Particle Filter-based Direct Visual Servoing

Quentin Bateux, Eric Marchand

Particle Filter-based Direct Visual Servoing

Quentin Bateux, Eric Marchand

Abstract— **With respect to classical visual servoing (VS) technics based on geometrical features, the main drawback of direct visual servoing is its limited convergence area. In this paper we propose a new direct visual servoing control law that relies on a particle filter to achieve non-local and non-linear optimization in order to increase this convergence area. Thanks to multi-view geometry and image transfer techniques, a set of particles (which correspond to potential camera velocities) are drawn and evaluated in order to evaluate the best camera trajectory. This new control law is validated on a 6 DOF positioning task performed on a real gantry robot and statistical comparisons are also provided from simulation results.**

I. Introduction

Visual servoing aims to control the dynamic of a system based on visual information provided by one or multiple cameras[4]. To achieve a positioning task, a visual servoing control law regulates an error in the image space. While the classical methods rely on extracting and tracking a set of geometrical features in the image, direct visual servoing [6][5] bypasses this matching and tracking process (which remain under broad investigation) by relying solely on the pixel intensities of the current and target images. Direct VS approaches thus directly rely on the information at hand with a minimal amount of image processing.

Building on this idea [6], various visual servoing control laws have been proposed based on global descriptors, such as mutual information [7] or histogram distances [3]. Although these methods added improvements to the robustness of the control schemes with respect to global changes such as illumination changes, these methods remained restricted in terms of convergence area due to the narrowness of the convex area of the associated cost functions (which is highly non-linear) around the desired positions. In order to tackle this issue, as for other complex optimization problems we proposed to rely on particle filters, also called Sequential Monte Carlo approaches. Since its first introduction in [10] this method proved to be a powerful tool, especially in the tracking field where it expanded from single object tracking [14], [12] to multitarget tracking [15]. This optimization method was also applied to robot localization problems, such as SLAM [9].

To take advantage on the robustness of the particle filter, this paper proposes a method to integrate a particle filter within a visual servoing control law. The main issue for a visual servoing control law is to establish a link between the 2D information and the 3D position of the robotic system. The general idea of particle filter techniques is to represent the required posterior density function by a set of random samples (particles) with associated weights and to drive the camera toward the optimum based on these samples and weights [2]. It is then important to be able to predict the image view from a virtual position. To achieve this goal we will rely on homographies [11] which allows the current image to be warped into a virtual image as it should be seen from a virtual camera position. After comparisons based on the sum of squared differences (SSD), the best particle (position) is then selected and the camera velocity to be sent to the robot is then computed in order to move the camera in the direction of this optimal particle (position). To illustrate the behaviour of the control, we chose a photometric visual servoing approach [6]. The main drawback of this method being the limited convergence area, the proposed particle-filter based control law demonstrated it efficiency. Let us note that other works that consider predictive control are also based on the prediction of a cost function estimated using generated points of view (eg, [1]) however our goals (increasing convergence area versus handling loss of visibility of extracted features) and the developed approach (particle filter versus predictive control) are clearly different.

This paper is structured as follows: the next section recalls the basics of direct visual servoing control law, Section III details the integration of a particle filter in a visual servoing control law and Section IV presents experimental results performed on a 6 DOF gantry robot, as well as a statistical comparisons between the proposed method over the direct visual servoing approach.

II. Classical direct visual servoing

The aim of a positioning task is to reach a desired pose of the camera $\mathbf{r}^*$, starting from an arbitrary initial pose. To achieve that goal, one needs to define a cost function that reflects, in the image space, this positioning error and that needs to be minimized. Given the actual pose of the camera $\mathbf{r}$ the problem can therefore be written as an optimization process:

$$\widehat{\mathbf{r}} = \arg\min_{\mathbf{r}} \rho(\mathbf{r}, \mathbf{r}^*) \tag{1}$$

where $\widehat{\mathbf{r}}$, the pose reached after the optimization process (servoing process), is the closest to $\mathbf{r}^*$ (optimally $\widehat{\mathbf{r}} = \mathbf{r}^*$).

To avoid extraction and tracking of geometrical features (such as points, lines, etc) [6][5] have introduced the notion of direct (or photometric) visual servoing where the visual feature is the image considered as a whole. In this case the feature is the image itself: $\mathbf{I}(\mathbf{r})$. This means that the optimization process is given by [5]:

$$\widehat{\mathbf{r}} = \arg\min_{\mathbf{r}} (\mathbf{I}(\mathbf{r}) - \mathbf{I}^*)^\top (\mathbf{I}(\mathbf{r}) - \mathbf{I}^*), \tag{2}$$

Quentin Bateux and Eric Marchand are with Université de Rennes 1, IRISA, Inria Rennes, Lagadic team, France, Quentin.Bateux@irisa.fr, Eric.Marchand@irisa.fr

where $\mathbf{I}(\mathbf{r})$ and $\mathbf{I}^*$ are respectively the image seen at the position $\mathbf{r}$ and the reference image (both N pixels).

In that case $\mathbf{L_I}$, the interaction matrix, that links the variation of $\dot{\mathbf{I}}$ to the camera velocity, can be expressed as:

$$\mathbf{L_I} = -\nabla \mathbf{I}^\top \mathbf{L_x}. \tag{3}$$

$\mathbf{L_x}$ is the interaction matrix of a point (see details in [5]).

This equation leads to the expression of the velocity that is applied to the robot. The control law is given by:

$$\mathbf{v} = -\lambda \mathbf{L_I^+}(\mathbf{I}(\mathbf{r}) - \mathbf{I}^*), \tag{4}$$

where λ is a positive scalar and $\mathbf{L_I^+}$ is the pseudo-inverse of the interaction matrix. Note that this direct visual servoing method is implemented in the ViSP library [13].

The strength of the direct visual servoing approach is to provide a clear mathematical framework that will fully exploit the convex properties of the SSD cost function around the desired position [6]. However, this cost function is highly non-linear and may be non-convex around the current camera position. In that case, this class of methods may not succeed in converging, either falling into a local minimum, or even diverging completely. This is mainly due to the use of a control law similar to a Newton optimization scheme. Therefore, in order to tackle this limitation, this paper presents an alternative approach for the optimization strategy, namely the Sequential Monte Carlo method, also called particle filter.

III. Particle Filter applied as visual servoing control law

The goal of this section is to propose a new visual servoing control scheme based on the particle filter (PF). We first explain the basics of the PF, and then propose a way to instantiate the general concept of particle in terms of visual servoing concepts.

A. Theoretical foundations of the PF

The main concept behind the PF set of methods is to perform the optimization of a given cost function by estimating it iteratively. It does so by computing a set of discrete estimations, where each element is a "particle". Then, according to the respective value of the estimated cost function associated to each particle, the PF algorithm updates its particles in order to optimize the global value of the estimates. The most interesting property of the PF is its ability to search a large space of solutions by not restraining itself to local estimation as the classical Newton-like optimization methods. This is linked to the fact that the particles set is free to explore the cost function as far as it can estimate its value, and the particles can be scattered in the search space enough to bridge the potential gaps of local minima, a property that a Newton-like optimization scheme does not possess since it rely solely on the local value of the cost function gradient.

PF is a general framework; multiple variations have been proposed over time, each of which exploits the same principles but with some changes regarding the update of the particles' parameters. Here we choose to apply the original version of the PF proposed in [10]. This approach is now widely referred to as *Sequential Importance Sampling (SIS)* and can be explained as follows:
Given a set of random particles

$$\{\mathbf{r}_{k-1}(i) : i = 1, ..., N_p\}, \tag{5}$$

where N_p is the number of particles in the set, obtained from the probability density function (PDF) $p(\mathbf{r}_{k-1}|\mathbf{y}_{k-1})$, where $\mathbf{y}_{k-1}$ is the set of measurements at time $k-1$ such that $\mathbf{y}_{k-1} = \{\mathbf{y}_{k-1}(i) : i = 1, ..., N_p\}$, where $\mathbf{y}_i = \rho(\mathbf{r}_i)$, $\rho(.)$ is the cost function to be optimized and where $\mathbf{r}$ is a full camera pose. The goal is to propagate and update these samples to get $\mathbf{x}_k(i)$ in order to get the distribution $p(\mathbf{r}_k|\mathbf{y}_k)$ that approximates the function $\rho(.)$. In order to get this approximation, the SIS algorithm performs the following three-steps:

- *Prediction step*: we obtain the samples at time k with $\mathbf{r}_k(i) = f(\mathbf{r}_{k-1}(i), w_{k-1}(i))$ where $f(\mathbf{r}_k(i))$ is the system transition function such as $\mathbf{r}_k = f(\mathbf{r}_{k-1}, \mathbf{w}_{k-1})$, and $w_k(i)$ is the weight associated to the sample i at time k. In this paper, we do not assume any displacement model for our system. In this case, the system transition function becomes a randomization function that diffuses the particles around their previous positions to keep the system from degenerating into having all particles at a single state (see 'Resampling step').
- *Update step*: Once we can compute the measurements $\mathbf{y}_k$, we can evaluate the likelihood of each particle and determine an associated weight $w_k(i)$. This likelihood is defined as:

$$w_k(i) = \frac{p(\mathbf{y}_{k-1}(i)|\mathbf{r}_{k-1}(i))}{\sum_{j=1}^{N_p} p(\mathbf{y}_{k-1}(j)|\mathbf{r}_{k-1}(j))} \tag{6}$$

- *Resampling step*: Once all the weights associated to the particles set have been computed, each of the particles is resampled by drawing a new particle from the PDF defined by the current set of weighted particles. In more practical terms, the resampling process will give every particle the state of another particle within the current particle set (including itself), with a chance proportional to the weight of each particle. The consequence is that statistically, the lower-weight particles are ruled out and new ones are created in their place near (in the state space) the higher-weight particle. This process can also be referred to as a condensation process [12]. This leads to a concentration of the particles set around the more satisfying zones of the search space.

By applying this algorithm, given that the initial state of particles overlap in a globally convex zone of the search space in terms of cost function value, the algorithm [6] converges toward an optimum, independently of the initial position of the gravity center of the set of particles.

B. PF-based visual servoing control law

As seen in the Section II, visual servoing can be viewed as an optimization problem. In this section we will see how PF

can be considered to replace the classical control law (which is similar to a Newton-like optimization technique).

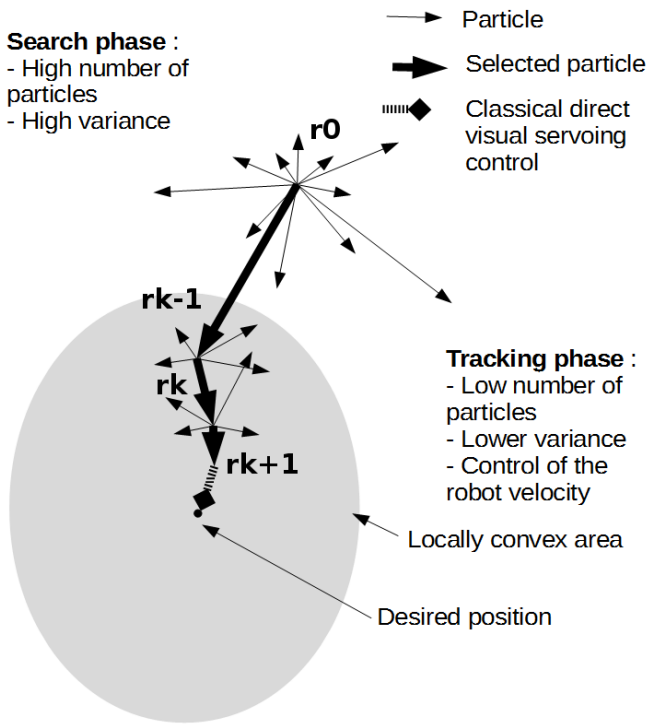


Fig. 1. Illustration of the proposed control scheme

The main idea is described in figure 1: from the current position we draw a set of particles which is just a set of possible camera positions reachable from the current camera position $\{\mathbf{r}_k(i), i = 1...N_p\}$. From these virtual camera positions (particles), a virtual image $I_k(i)$ can be simulated (see Section III.B.1), and the weight $w_k(i)$ that measures the quality of each particle (i.e., of each potential camera position) is computed according to the quality criterion ρ defined, in our case by the SSD (see Section III.B.2). The best particle is then selected, and the camera velocity to be sent to the robot is computed in order to move the camera in the direction of this optimal particle (position). The particles are then resampled, and the process is iterated until we reach the desired position.

In order to consider such a control scheme, a few points have to be defined:

1) the search space and how to generate particles within this search space.
2) an evaluation of the quality of a particle (i.e., estimation of $w_k(i)$ in equation (6)
3) the camera velocity

Let us now examine these three processes.

1) Search space and particles generation: We want to control the 6 DOF of the robot; it seems natural then to define the search space as the $SE(3)$ space. Let us define $\mathcal{F}_k$ as the camera frame at iteration k of the positioning process. Each generated particle $\mathbf{r}_k(i), i = 1...N_p$ is a virtual camera position that is defined by a position expressed in $\mathcal{F}_k$. $\mathbf{r}_k(i)$ is then a coordinate vector $\mathbf{r}_k(i) = (tx, ty, tz, rx, ry, rz)$ that defines both the translational and the rotational positions of the particle in the camera frame. Alternately, one can denote this position by a homogeneous matrix ${}^{k(i)}\mathbf{T}_k$ defined as:

$$ {}^{k(i)}\mathbf{T}_k = \begin{pmatrix} {}^{k(i)}\mathbf{R}_k & {}^{k(i)}\mathbf{t}_k \\ \mathbf{0}_{3\times 1} & 1 \end{pmatrix}, \quad (7) $$

where ${}^{k(i)}\mathbf{R}_k$ and ${}^{k(i)}\mathbf{t}_k$ are the rotation matrix and translation vector respectively that define the position of the particle in the current camera frame $\mathcal{F}_k$.

Since we work with 2D information only, we do not accurately know the depth information between the camera and the scene. This can prove to be problematic since the projection $\mathbf{r}_k(i)$ uses depth information. In case of strong violation of this hypothesis, it may cause a strong discrepancy between the predicted image and the actual image seen from a particular viewpoint. This can result in a camera motion that takes out of the camera field of view most of the usable visual information and therefore causes the system to diverge. This is illustrated in figure 2, where the black camera represents the positioning that would be reached by the robotic system, whereas the predicted image is simulated as being seen from the blue camera. As illustrated, this offset in position and orientation can potentially lead to taking most of the useful image information out of the camera view.

In order to alleviate this issue and allow the system to compensate for this error with translations during all iterations rather than accumulating it as rotations offsets, we perform a change in the transformation matrix ${}^{k(i)}\mathbf{T}_k$ to compute the position of the virtual camera in camera reference frame. The idea is that, in order to limit the offsets in rotation that are easily harmful, we perform the rotation part of the transformation in the image plane reference frame. The effect of this change is illustrated in the figure 3 which applies the same magnitude of rotation as in figure 2, but this time with the change in reference frame. We can see that the leverage effect disappears and that the center of the initial image is still at the center of the warped image; the offset due to the error in depth being transferred mainly into the translation along the optical axis, inducing a translation along the z-axis, which can be easily compensated by the closed loop.

This change has two main consequences:

- Stronger rotations can be performed without risking transferring the original image outside the virtual camera view, since without leverage effect, the transfer no longer displaces the center of the warped image in the predicted view when performing rotations;
- In order to keep the projection aligned with the desired view, the PF will select particles with a compensation in the translations to make up for the case where the camera is not facing the center of the desired image.

This latter point has important consequences regarding the behaviour of the system as the drift will be kept minimal since the PF will lead the system to keep facing toward the desired part of the image as much as possible while acting on the translations rather than on the rotations.

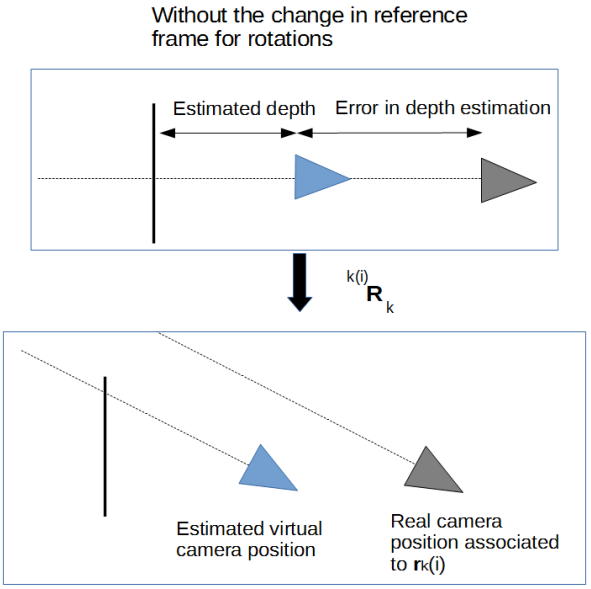


Fig. 2. Applying a pure rotation transformation without changing the reference frame with badly estimated depth results in an inadequate camera orientation

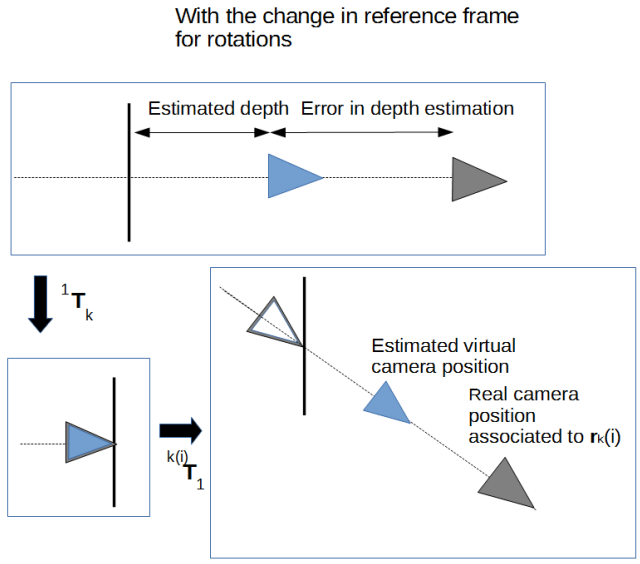


Fig. 3. Apply a pure rotation transformation with a change in the reference frame with badly estimated depth conserves the expected camera orientation

The expression of ${}^{k(i)}\mathbf{T}_k$ associated with this transformation is:

$$ {}^{k(i)}\mathbf{T}_k = {}^{k(i)}\mathbf{T}_1 \, {}^{1}\mathbf{T}_k + \frac{{}^{k(i)}\mathbf{t}_k}{d}\mathbf{n}^\top \tag{8} $$

where $\mathbf{n}$ and d are the normal and distance to the origin of the scene plane expressed in camera frame $\mathcal{F}_{(k)}$ and where ${}^{k(i)}\mathbf{T}_1$ is an intermediary transformation matrix defined such as:

$$ {}^{k(i)}\mathbf{T}_1 = \begin{pmatrix} \mathbf{I}_{3\times 3} & \mathbf{n}_{rotated} \\ \mathbf{0}_{3\times 1} & 1 \end{pmatrix}, \tag{9} $$

where $\mathbf{n}_{rotated}$ is the normal vector $\mathbf{n} = (n_x, n_y, -1)^\top$ associated with the image plane rotated as:

$$ \mathbf{n}_{rotated} = {}^{k(i)}\mathbf{R}_k \mathbf{n} \tag{10} $$

And ${}^{1}\mathbf{T}_k$ is a second intermediary transformation matrix defined such as:

$$ {}^{1}\mathbf{T}_k = \begin{pmatrix} 1 & 0 & 0 & n_x \\ 0 & 1 & 0 & n_y \\ 0 & 0 & 1 & dn_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{11} $$

In order to explore the search space properly (both the local and farther neighborhood), we define the system transition function f_k as a Gaussian function. $f(\mathbf{x})$ can then be expressed as:

$$ f(\mathbf{x}) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(\mathbf{x}-\mu)^2}{2\sigma^2}} \tag{12} $$

With σ^2 the variance and μ the expected value. It has to be noted that as this function's role is to diffuse each particle around its own position, we always have $\mu = 0$.

2) Particle evaluation: We now need to devise a feasible way of computing the weight of each particles. However it is not feasible to actually move the robot around the 3D space to collect image samples in order to weight the particles, due to considerations both in terms of speed and practicality. Therefore, predicting the image that would be taken from each particle position proves to be crucial for such system.

As stated in Section II-A, a key element of the PF is to estimate the value of the weight $w_k(i)$ associated with each particle. This weight, which reflects the particle quality, is computed according to equation (6). Let us denote $\mathbf{I}_k$ as the current image and $\mathbf{I}(\mathbf{r}_k(i))$ as the predicted image acquired from the virtual position defined by particle $\mathbf{r}_k(i)$ in the current camera frame. According to equation (6) the weights $w_k(i)$ are functions of $p(\mathbf{y}_k(j)|\mathbf{x}_k(j))$.

Since we use the SSD cost function expressed in equation (2), then:

$$ \begin{aligned} p(\mathbf{y}_k(j)|\mathbf{x}_k(j)) &= \rho\big(\mathbf{I}(\mathbf{r}_k(i), \mathbf{I}^*)\big) \\ &= (\mathbf{I}(\mathbf{r}_k(i) - \mathbf{I}^*)^\top (\mathbf{I}(\mathbf{r}_k(i) - \mathbf{I}^*) \end{aligned} \tag{13} $$

where $\mathbf{I}^*$ is the image taken at the desired position.

When the similarity between these two images increases, the weight associated with this particle will be important, driving the PF towards its position.

The main issue is then to predict the image $\mathbf{I}(\mathbf{r}_k(i))$ using the current image $\mathbf{I}_k$. The chosen solution relies on image transfer techniques [11]. When a general motion is considered, it is not possible to predict this image *a priori* when only $\mathbf{I}_k$ is known. Nevertheless, if one assumes that the scene is planar, a homography [11] can be considered to achieve this prediction.

Indeed, assuming that the position of the particle in the camera is given by ${}^{k(i)}\mathbf{T}_k$ as seen in the previous paragraph, the homography that will transfer image $\mathbf{I}_k$ on image $\mathbf{I}(\mathbf{r}_k(i))$ is given by:

$$ {}^{k(i)}\mathbf{H}_k = {}^{k(i)}\mathbf{R}_k + \frac{{}^{k(i)}\mathbf{t}_k}{d}\mathbf{n}^\top, \tag{14} $$

where $\mathbf{n}$ and d are the normal and distance to the origin of the scene plane expressed in camera frame $\mathcal{F}_{(k)}$. This means that a pixel $\mathbf{x}_k$ in $\mathbf{I}_k$ will be warped at coordinates $\mathbf{x}_{k(i)}$ in $\mathbf{I}(\mathbf{r}_k(i))$ via a homography ${}^{k(i)}\mathbf{H}_k$ such that

$$\mathbf{x}_{k(i)} = {}^{k(i)}\mathbf{H}_k \ \mathbf{x}_k. \tag{15}$$

Using equation (15), it is easy to synthesize virtual images $\mathbf{I}(\mathbf{r}_k(i)), i = 1...N_s$ for the whole set of particles as depicted in figure 4.

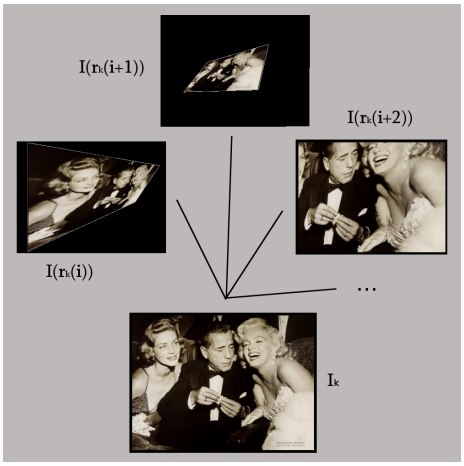


Fig. 4. Generation of examples of transferred images though homography to predict the view from virtual camera positions

3) PF-based control law: The general scheme that defines the control law can be expressed as follows:

- Initialization of the PF with a large variance in order to discover the convex zone;
- At the start of the control loop (iteration 0):
  - perform an iteration of the PF: draw N_p particles $\mathbf{r}_k(i)$ (as described in Section III.B.1) and compute their weights $w_k(i)$ according to equations (6) and (13). From these particles, select the one with the highest weight: $\mathbf{r}_k(best)$
  - send a velocity command to the robot in the direction of this particle (position):

$$\mathbf{v} = \lambda \ \mathbf{r}_{k(best)} \ \rho\big(\mathbf{I}_k, \mathbf{I}^*\big) \tag{16}$$

    where $\mathbf{r}_{k,best}$ is the position of the particle of highest weight in the current camera frame and λ is a gain that determines the amplitude of the system velocity. The weighting by the current measure of the cost function allows the velocity to decrease evenly when approaching the optimal position.
- When the estimated cost of the highest-weighted particle and the current SSD cost are equal, we consider that we are close enough to the optimal position and we switch to the classical SSD-based control law [6] presented in Section II. This provides the benefit of the sub-pixel positioning accuracy of this approach by allowing us to overcome the discretization error of the PF that would lead to a less precise final position. An alternative solution would be to increase greatly the number of particles, but this proves to be prohibitive in terms of computational costs.

C. Practical details on our Particle Filter implementation for visual servoing

As we aim to apply this control scheme on a dynamic real-time system, some adaptations have been performed in order to build on the *a-priori* knowledge concerning visual servoing tasks:

- around the desired position, the cost function exhibits a locally convex area;
- the warped image generated by homography needs to contain enough common information with the desired image to be considered relevant.

In order to keep the computational cost low enough for real-time application, we decided to perform a two step process: we start with an exploration phase with 1000 particles generated and a large variance in $f(\mathbf{x})$. This increases greatly the exploration of the search-space at the cost of a couple of seconds delay before starting the robot motion. At the end of this first phase, we sample only 100 particles from the previous particle set and the variance is reduced to keep a good density of particles. We then apply the process that as been described the previous section.

In the current implementation, the variance parameters are set empirically in order to generate a low number of out-of-view images (illustrated by image B in figure 5), the latter being also penalized in terms of particle weight by a rejection ratio that penalizes images that feature low levels of information (but which can still have a decent SSD cost despite being off due to the nature of the SSD metric) so that they cannot be selected and drive the system to divergence. The amount of information is easily measured at warping time by measuring the amount of background (unknown parts of the scene) in the warped image.

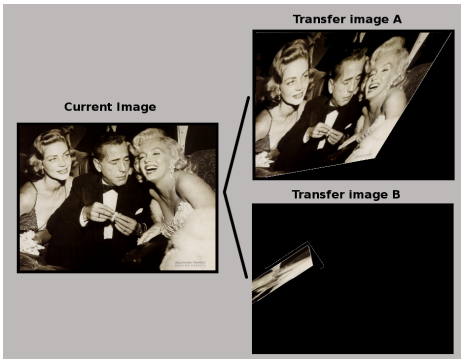


Fig. 5. Warping an image into a suitable image and an uninformative one

IV. Experimental validation

A. Validation on a 6 DOF positioning task on Gantry robot

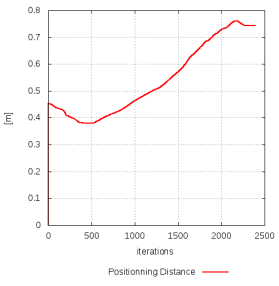
In order to validate this approach, we performed an experiment on a 6 DOF robot. In figure 7 we can see the results of the experiment: figure 7(f) shows the image seen

from the starting position of the camera, and figure 7(g) shows the image taken at the end of the motion. In order to visualize the error, the difference between these images and the desired image can be seen in figure 7(h) and figure 7(i) respectively. Since we use the SSD between these two images as our cost function, this error image is directly linked to the error measure that is visualized on figure 7(b). The error in the positioning of the camera through the experiment can be seen in figure 7(a) and this metric is computed from the sum of the absolute values of the pose errors in translations (figure7(c)). The rotational errors can be seen in figure 7(d). Lastly, figure 7(e) displays the evolution of the velocities applied on all 6 DOF of the robot. The initial pose consists of an offset from the initial pose of (-10cm, -40cm, 18cm, -52°, 2°, 10°), with a depth value of 80cm at the desired position. The final recorded position error is less than 1mm.

The experiment consists of resolving the presented positioning problem by applying the proposed PF-based control law in order to deal with the strong initial error, especially in terms of rotation, which has a significant effect on the SSD error value and makes impossible to solve by the traditional direct SSD-based visual servoing (at this starting position, the cost function is non-convex and highly non-linear and therefore cannot be optimized properly with Gauss-Newton related methods that rely on such properties), as seen in figure 6 where we try unsuccessfully to solve this same positioning task with the direct visual servoing method presented in Section II.

We can see that the proposed control scheme succeeds in converging efficiently toward the desired position, as seen in figure 7(a) with a continuous decrease in positioning error.

At the end of the motion (here at the iteration 1404), when the offset between the best predicted error and the currently recorded error is small enough, we switch to the direct visual servoing control in order to reach the desired position with a sub-millimetric precision.

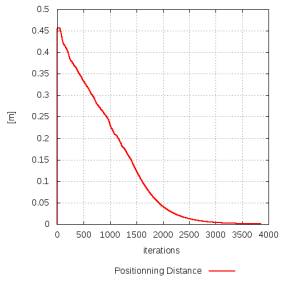


a

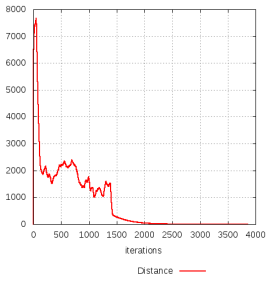
b

c

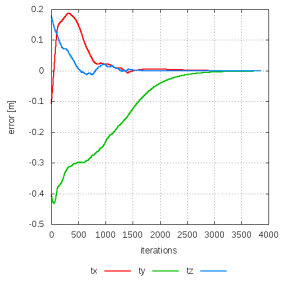
Fig. 6. Direct visual servoing. (a) Positionning error. (b) Initial image. (c) I at the end of the motion.



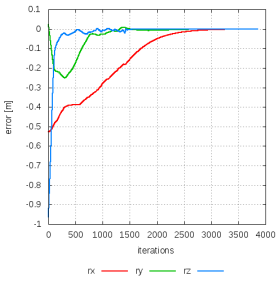
a



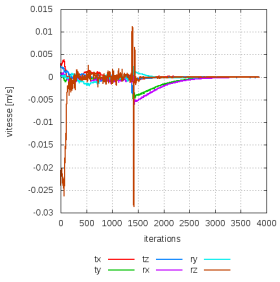
b



c



d



e

f

g

h

i

Fig. 7. Hybrid visual servoing, first PF-based, then by classical direct VS control (switch at iteration 1404). (a) Positionning error. (b) SSD distance. (c) Translational errors. (d) Rotational errors. (e) Camera velocities in m/s and rad/s. (f) Initial image. (g) I at the end of the motion. (h) I - I* at initial position. (i) I - I* at the end of the motion

B. Statistical comparison between PF-based and direct visual servoing in simulated environment

Since this new method is validated on a real robot experiment, further investigation is required to assert the advantage in terms of increased convergence area. To this end, we performed a statistical analysis of the convergence successes in a simulated environment. The process is the following: we define a desired pose and its associated image, and then we perform several trials with randomized initial positions around the desired position. These random positions are created by adding a Gaussian noise to the desired position. By increasing this noise's variance, it is possible to smoothly increase the difficulty of the positioning task. Figure 8 shows the result of this process with 10 increases in the spatial noise variances. 40 runs are performed to get the success rate percentage at each noise iteration. The distance from the camera to the image plane has been set to 20cm and the noise's variances are such as:

- from 0 to 4cm for the x/y translations;
- from 0 to 2cm for the z translation;
- from 0 to 10° for the rotations around the x/y axis;
- from 0 to 50° for the rotation around the z axis.

The rotation transformations being the most difficult to handle.

We can clearly see from the resulting curves that the proposed method has much better performances than the direct visual servoing method in terms of convergence radius, especially when strong rotations around the optical axis are involved.

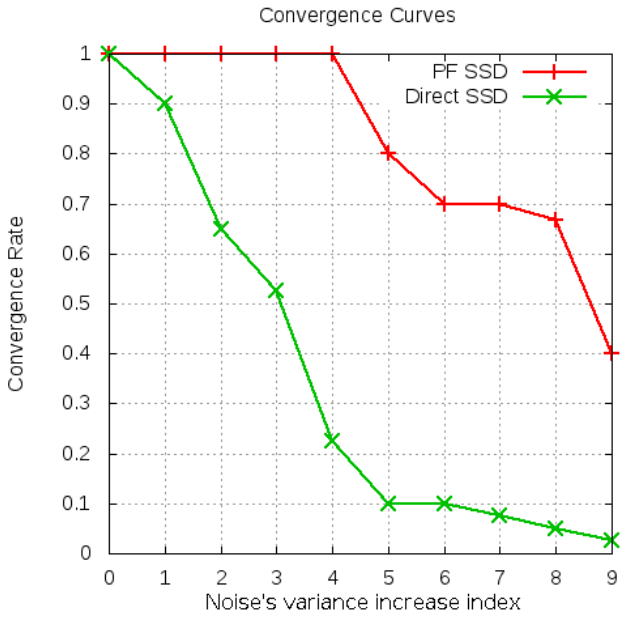


Fig. 8. Convergence rates of direct visual servoing methods with PF-based and classic control laws

V. Conclusion and perspectives

In this paper we presented a novel visual servoing control scheme based on a particle filter. We presented a way to represent a particle as a virtual camera position through the use of a warp function based on homography in order to predict the view of this camera and evaluate the associated particle by computing an SSD measure. This experiment showed that the integration of the particle filter improves the convergence area of the control law.

Future works concerns the application of this technique to other more elaborate costs functions such as histograms of oriented gradients or mutual information that have been proven to be robust global descriptors and adequate for performing visual servoing tasks [3][8].

References

[1] G. Allibert, E. Courtial, and F. Chaumette. Predictive control for constrained image-based visual servoing. *IEEE Trans. on Robotics*, 26(5):933–939, 2010.

[2] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Trans on Signal Processing*, 50(2):174–188, 2002.

[3] Q. Bateux and E. Marchand. Histograms-based visual servoing. *IEEE Robotics and Automation Letters*, 2(1):80–87, January 2017.

[4] F. Chaumette and S. Hutchinson. Visual servo control, Part I: Basic approaches. *IEEE Robotics and Automation Magazine*, 13(4):82–90, December 2006.

[5] C. Collewet and E. Marchand. Photometric visual servoing. *IEEE Trans. on Robotics*, 27(4):828–834, August 2011.

[6] C. Collewet, E. Marchand, and F. Chaumette. Visual servoing set free from image processing. In *IEEE Int. Conf. on Robotics and Automation, ICRA'08*, pages 81–86, Pasadena, CA, May 2008.

[7] A. Dame and E. Marchand. Mutual information-based visual servoing. *IEEE Trans. on Robotics*, 27(5):958–969, October 2011.

[8] Amaury Dame. *A unified direct approach for visual servoing and visual tracking using mutual information*. PhD thesis, Rennes 1, 2010.

[9] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping : Part i. *IEEE Robotics & Automation Magazine*, 13(2):99–108, 2006.

[10] N. Gordon, D. Salmond, and A. Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. In *IEE Proceedings Radar and Signal Processing*, volume 140, pages 107–113, 1993.

[11] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2001.

[12] M. Isard and A. Blake. Condensation – conditional density propagation for visual tracking. *Int. J. Computer Vision*, 29(1):5–28, January 1998.

[13] E. Marchand, F. Spindler, and F. Chaumette. ViSP for visual servoing: a generic software platform with a wide class of robot control skills. *IEEE Robotics and Automation Magazine*, 12(4):40–52, December 2005. Special Issue on "Software Packages for Vision-Based Control of Motion", P. Oh, D. Burschka (Eds.).

[14] K. Nummiaro, E. Koller-Meier, and L. Van Gool. Object tracking with an adaptive color-based particle filter. In *Pattern Recognition*, pages 353–360. 2002.

[15] K. Okuma, A. Taleghani, N. De Freitas, J. Little, and D. Lowe. A boosted particle filter: Multitarget detection and tracking. In *European Conf. on Computer Vision*, pages 28–39. 2004.